UDACITY

‹ Back to Data Analyst Nanodegree

Explore US Bikeshare Data

审阅

代码审阅 4

HISTORY

▾ bikeshare.py 4

```
import time
import pandas as pd
import numpy as np

CITY_DATA = { 'chicago': 'chicago.csv',
              'new york': 'new_york_city.csv',
              'washington': 'washington.csv' }

def get_filters():
    """
    Asks user to specify a city, month, and day to analyze.

    Returns:
        (str) city - name of the city to analyze
        (str) month - name of the month to filter by, or "all" to apply no mont
        (str) day - name of the day of week to filter by, or "all" to apply no
    """
    print('Hello! Let\'s explore some US bikeshare data!')

    # get user input for city (chicago, new york city, washington).
    city = get_item('Would you like to see data for Chicago, New York, or Washi
             'Your answer is not correct, please try again.\n',
             CITY_DATA.keys(),
             lambda x: str.lower(x))
    print('Looks like you want to hear about %s! If this is not true, restart t

    # get user input for preferences of time filter
    user_filter = get_item('Would you like to filter the data by month, day, bc
             'Your answer is not correct, please try again.\n',
             ['month', 'day', 'both', 'none'],
             lambda x: str.lower(x))
    print('We will make sure to filter by %s!\n\n' % user_filter)
    # month can be assigned
    month = 'all'
    if user_filter == 'month' or user_filter == 'both':
        # get user input for month (all, january, february, ... , june)
        months = {'january': 1, 'february': 2, 'march': 3, 'april': 4, 'may': 5
        month_key = get_item('Which month? January, February, March, April, May
            'Your answer is not correct, please try again.\n',
            months.keys(),
            lambda x: str.lower(x))
        month = months[month_key]
        print('You will make sure to choose %s!\n\n' % month_key)
    # day can be assigned
    day = 'all'
    if user_filter == 'day' or user_filter == 'both':
        # get user input for day of week (all, monday, tuesday, ... sunday)
        days = {'sunday': 0, 'monday': 1, 'tuesday': 2, 'wednesday': 3, 'thursd
        day_key = get_item('Which day? Sunday, Monday, Tuesday, Wednesday, Thur
            'Your answer is not correct, please try again.\n',
            days.keys(),
            lambda x: str.lower(x))
        day = days[day_key]
        print('You will make sure to choose %s!\n\n' % day_key)
```

棒极了

很好的使用了函数减少了程序的冗余，并将逻辑封装了起来，做的很好。相信你也从

建议可以试一试挑战一下关于视频中的模糊输入的功能，可以使用 `difflib` 库进行，下面的示例可以供你作为参

```
import difflib
def compare_2_string(str1,str2):
    inner_counter = 0
    for i, s in enumerate(difflib.ndiff(str1, str2)):
        if s[0] in ['+', '-']:
            inner_counter += 1
    return inner_counter

def get_item(input_print, error_print, enterable_list, get_value):
    while True:
        ret = get_value(input(input_print))
        dict_like = {}

        #计算输入和列表项二者的差别，并记录到字典中
        for enterable_item in enterable_list:
            #这里使用除法可以防止字符串长度导致长字符串较难被匹配上
            dict_like[enterable_item] = compare_2_string(ret,enterable_item)/len(e

        #对计算结果进行排序
        top = sorted(dict_like.items(),key=lambda x:x[1])[0]

        #如果输入和列表相同，则直接返回结果
        if top[1] == 0:
            return ret
        #否则等待用户进行确认
        else:
            confirm = input(error_print + "\nBut may you want {}, enter y to confi
(top[0]))
            if confirm.lower() == 'y':
                return top[0]

       print('-'*40)
       return city, month, day

   def get_item(input_print, error_print, enterable_list, get_value):
       """
       Get input in the same pattern.

       Returns:
           (str) ret - the input of the user in certain pattern
       """
```

棒极了

很好的对函数进行了注释，做的好。

```
    while True:
        ret = input(input_print)
        ret = get_value(ret) # 这里执行作为参数的函数
        if ret in enterable_list:
            return ret
        else:
            print(error_print)

def load_data(city, month, day):
    """
    Loads data for the specified city and filters by month and day if applicabl

    Args:
        (str) city - name of the city to analyze
        (str) month - name of the month to filter by, or "all" to apply no mont
        (str) day - name of the day of week to filter by, or "all" to apply no
    Returns:
        df - Pandas DataFrame containing city data filtered by month and day
    """
    # Loads data for the specified city
    try:
        df = pd.read_csv(CITY_DATA[city])
    except Exception as e:
        print('The program encountered an error: ', e)
        exit()

    # convert the Start Time column to datetime
    df['Start Time'] = pd.to_datetime(df['Start Time'])
    # extract month from the Start Time column to create an month column
    df['month'] = df['Start Time'].dt.month
    # extract day from the Start Time column to create an day column
    df['day'] = df['Start Time'].dt.weekday
    # extract hour from the Start Time column to create an hour column
    df['hour'] = df['Start Time'].dt.hour

    # filter some rows based on user's filter
    if month == 'all' and day == 'all':
        return df
    elif month == 'all' and day != 'all':
        return df[df.day == day]
    elif month != 'all' and day == 'all':
        return df[df.month == month]
    else:
        return df[(df.month == month) & (df.day == day)]

def time_stats(df, month_is_all, day_is_all):
    """Displays statistics on the most frequent times of travel."""

    print('\nCalculating The Most Frequent Times of Travel...\n')
    start_time = time.time()

    if month_is_all:
        # display the most common month
        popular_month = df['month'].mode()[0] # find the most popular month
        month = ['', 'January', 'February', 'March', 'April', 'May', 'June']
        print('Most popular month: %s\n' % month[popular_month])

    if day_is_all:
        # display the most common day of week
        popular_day = df['day'].mode()[0] # find the most popular week
        day = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'
        print('Most popular day of week: %s\n' % day[popular_day])

    # display the most common start hour
    popular_hour = df['hour'].mode()[0] # find the most popular hour
    print('\nMost popular hour: %d\n' % popular_hour)

    print("\nThis took %s seconds." % (time.time() - start_time))
    print('-'*40)


def station_stats(df):
    """Displays statistics on the most popular stations and trip."""

    print('\nCalculating The Most Popular Stations and Trip...\n')
    start_time = time.time()

    # display most commonly used start station
    popular_start_station = df['Start Station'].mode()[0] # find the most popul
    print('Most popular start station: %s\n' % popular_start_station)
    # display most commonly used end station
    popular_end_station = df['End Station'].mode()[0] # find the most popular e
    print('Most popular end station: %s\n' % popular_end_station)
    # display most frequent combination of start station and end station trip
    popular_trip = (df['Start Station'] + ' --> ' + df['End Station']).mode()[0
    print('Most popular trip: %s\n' % popular_trip)

    print("\nThis took %s seconds." % (time.time() - start_time))
    print('-'*40)

def trip_duration_stats(df):
    """Displays statistics on the total and average trip duration."""

    print('\nCalculating Trip Duration...\n')
    start_time = time.time()

    # display total travel time
    total = df['Trip Duration'].sum()
    print('\nTotal travel time: %f seconds\n' % total)

    # display mean travel time
    mean_val = df['Trip Duration'].mean()
    print('Mean travel time: %f seconds\n' % mean_val)
```

棒极了

很好的使用了库函数进行了数据的统计，做的好。

```
    print("\nThis took %f seconds." % (time.time() - start_time))
    print('-'*40)


def user_stats(df):
    """Displays statistics on bikeshare users."""

    print('\nCalculating User Stats...\n')
    start_time = time.time()

    # Display counts of user types
    print('\nCalculating statistics...\n\n')
    print('What is the breakdown of users?\n')
    user_types = df['User Type'].value_counts()
    print(user_types)

    # Display counts of gender
    print('\nCalculating statistics...\n\n')
    print('What is the breakdown of gender?\n')
    if 'Gender' in df.columns:
        gender = df['Gender'].value_counts()
        print(gender)
    else:
        print('No gender data to share.\n')

    # Display earliest, most recent, and most common year of birth
    print('\nCalculating statistics...\n\n')
    print('What is the oldest, youngest, and most popular year of birth, respec
    if 'Birth Year' in df.columns:
        print('Earliest year of birth: %d\n' % min(df['Birth Year'].dropna()))
        print('Recent year of birth: %d\n' % max(df['Birth Year'].dropna()))
        print('Most common year of birth: %d\n' % df['Birth Year'].dropna().mod
    else:
        print('No birth year data to share.\n')

    print("\nThis took %s seconds." % (time.time() - start_time))
    print('-'*40)


def main():
    while True:
        city, month, day = get_filters()
        df = load_data(city, month, day)
        time_stats(df, month == 'all', day == 'all')
        station_stats(df)
        trip_duration_stats(df)
        user_stats(df)

        restart = input('\nWould you like to restart? Enter yes or no.\n')
        if restart.lower() != 'yes':
            break
```

建议

虽然给的模板是这样的，但是实际输入为yes和no以外的情况还是会被当作no，所以建议试着用while或者for循环
须的，所以有很多有趣的问题留给你去自主探索。

```

if __name__ == "__main__":
    main()

```

▸ readme.txt

返回 PATH

学员 FAQ